Joni Jokinen

Uhkavaara-pelin toteuttaminen

Opinnäytetyö

Tietotekniikan koulutusohjelma

Toukokuu 2016

KYAMK
University of Applied Sciences

Tekijä/Tekijät	Tutkinto	Aika
Joni Jokinen	Tietotekniikka/ Peliohjelmointi	Toukokuu 2016

Opinnäytetyön nimi

Uhkavaara-pelin toteuttaminen — 33 sivua

Toimeksiantaja

Kaakkois-Suomen Pelastusalanliitto

Ohjaaja

Lehtori Niina Salmi

Tiivistelmä

Uhkavaara on iPad-alustalle suunniteltu peli. Peli on tarkoitettu 6 – 8-vuotiaille lapsille ja sen tarkoituksena on valistaa lapsia kodin ja ympäristön vaaroista. Uhkavaara toteutettiin käyttäen Unity-pelimoottoria C#-kielellä. Ohjelmoiminen aloitettiin 24.6.2015. Tätä ennen Kymenlaakson ammattikorkeakoulun pelisuunnittelun opiskelijat olivat tehneet valmiiksi suunnitelmat, joiden pohjalta peliä rakennettiin.

Opinnäytetyö kuvaa Unityn käyttämistä pelin rakentamiseen, itse projektin etenemistä, pelin siirtämistä Applen AppStoreen sekä tarkastelee lopputulosta. Monia Unityn ominaisuuksia on esitelty toiminnallisuuden ja tähän projektiin soveltamisen kannalta. Pelin tekemisen yhteydessä esiintyneisiin haasteisiin löydetyt ratkaisut on esitetty opinnäytetyössä. Myös opinnäytetyön aikana opittuja asioita ja kehitystä pohditaan.

Lopputuloksena aikaan saatiin suunnitelmien mukainen peli, joka ladattiin AppStoreen. Peliin suunnitellut ominaisuudet saatiin sisällytettyä lopulliseen versioon. Pelin tekeminen opetti paljon hyviä käytäntöjä, kuten muistinhallintaa ja tabletin koon huomioon ottamisen painikkeita asetellessa.

Asiasanat
Unity, C#

Author (authors)	Degree	Time
Joni Jokinen	Bachelor of Engineering	May 2016

Thesis Title

Development of the Game Uhkavaara — 33 pages

Commissioned by

Kaakkois-Suomen Pelastusalanliitto

Supervisor

Niina Salmi, Senior Lecturer

Abstract

Uhkavaara is a game developed for the iPad platform. The game is aimed at children from ages 6 to 8 and its purpose is to educate children of dangers of the environment and home. Uhkavaara was developed with the Unity game engine using C# as the programming language of choice. Programming started on the 24th of June 2015. Before this date the game design students of the Kymenlaakso University of Applied Sciences had created designs for the game.

This thesis describes the using of Unity for creating the game, its uploading to the AppStore and reviews the final product. Many features of Unity are described both by their functionality and by the way they were applied in this project. The solutions of the challenges are also explained. In addition, some of the lessons learned are documented.

The final result followed the plans accurately, and was succesfully sent to AppStore. The features that were originally planned were implemented into the game. Making the game taught the author about useful practices, such as memory usage and placing UI elements while keeping the dimensions of the target platform in mind.

Keywords

Unity, C#

SISÄLLYS

TERMIT JA LYHENTEET

AppStore	Applen omistama digitaalista sisältöä jakava alusta, josta Applen valmistamilla laitteilla pystyy lataamaan tai ostamaan sovelluksia laitteelle.
C#	Pelin ohjelmoimiseen käytetty ohjelmointikieli
Frame	Päätelaitteelle piirrettävä kuva pelistä, joka vaihtuu nopeasti.
iTunes Connect	Alusta, jonka avulla Applen laitteille kehitettäviä sovelluksia hallitaan.
Peliobjekti	Osa, jotka yhdessä koostavat pelin. Kaikki pelissä näkyvät osat ovat peliobjekteja: Pelihahmo, taustalla oleva grafiikka ja käyttöliittymän nappulat.
Scene	Erilaisia peliobjekteja sisältävä kokonaisuus Unityssä. Scenet tarkoittavat pelin eri osia, kuten päävalikkoa, tason valintaa ja pelattavaa osuutta.
Sprite	Yksi kuva, joka piirretään ruudulle pelissä.
Spritesheet	Monesta spritestä koostuva kuva, joka jaetaan yksittäisiin spriteihin Unityssä.
Tweening	Prosessi, joka muuttaa arvoa nykyisestä toiseen arvoon tietyn ajan kuluessa. Esimerkiksi peliobjektin sijaintia x-akselilla liikuttaen sitä eteenpäin, kunnes se on halutussa sijainnissa.
UI/GUI	(Graphical) User Interface eli pelin käyttöliittymä.
Unity	Pelimoottori, jonka avulla opinnäytetyönä tehty peli toteutettiin
Xcode	Applen kehittämä ohjelmistoympäristö, jonka avulla pystyy tekemään sovelluksia Applen laitteille. Xcoden avulla pystyy siirtämään Unityn avulla luodun pelin AppStoreen.

1 JOHDANTO

Opinnäytetyön tilaaja on Kaakkois-Suomen Pelastusalanliitto ry. Se on järjestö, joka toimii turvallisuusalalla. Järjestön toimintaan kuuluu muun muassa kansalaisten valistaminen vaaratilanteissa toimimiseen ja niiden välttämiseen. (1.) Liitto järjestää koulutuksia ja neuvontaa.

Pelastusalanliitto oli suunnitellut turvallisuuspeliä lapsille. Kymenlaakson ammattikorkeakoulun pelisuunnitteluopiskelijat olivat suunnitelleet pelin osat valmiiksi ja rahoitus oli saatu palosuojelurahastolta. Projektilta puuttui sopivasti ohjelmoija. Työ kuulosti sopivan haastavalta, mutta mahdolliselta, joten se otettiin opinnäytetyön aiheeksi.

Opinnäytetyön lopussa tarkoituksena on olla valmiina noin 6 - 8-vuotiaille lapsille suunnattu peli, joka valistaa erilaisista kodin ja ympäristön vaaroista. Pelissä on neljä erilaista tasoa, joissa pelaaja liikuttaa hahmoa etsien vaaroja ympäristöstä. Vaarat esitetään kysymysten muodossa. Pelissä on myös paloasema, joka sisältää kolme erilaista minipeliä: muistipelin, yhdistämispelin sekä tietovisan.

Projekti tehdään yhteistyönä pelisuunnittelun opiskelijoiden kanssa, jotka tuottavat peliin grafiikat ja suunnitelmat, joiden mukaan ohjelmointi toteutetaan. Peliäänet tilataan ulkopuoliselta ammattilaiselta.

2 SUUNNITTELU

Peli suunnitellaan valmiiksi, jotta kaikilla on selvä tavoite. Kun jokainen tietää, mitä tekee ja aikataulu on selvä, niin työ etenee sujuvasti. Koko tiimi suunnitteli aikataulun yhdessä. Jokainen arvioi omalla vastuulla olevien tehtävien keston, minkä pohjalta rakennettiin joustava aikataulu.

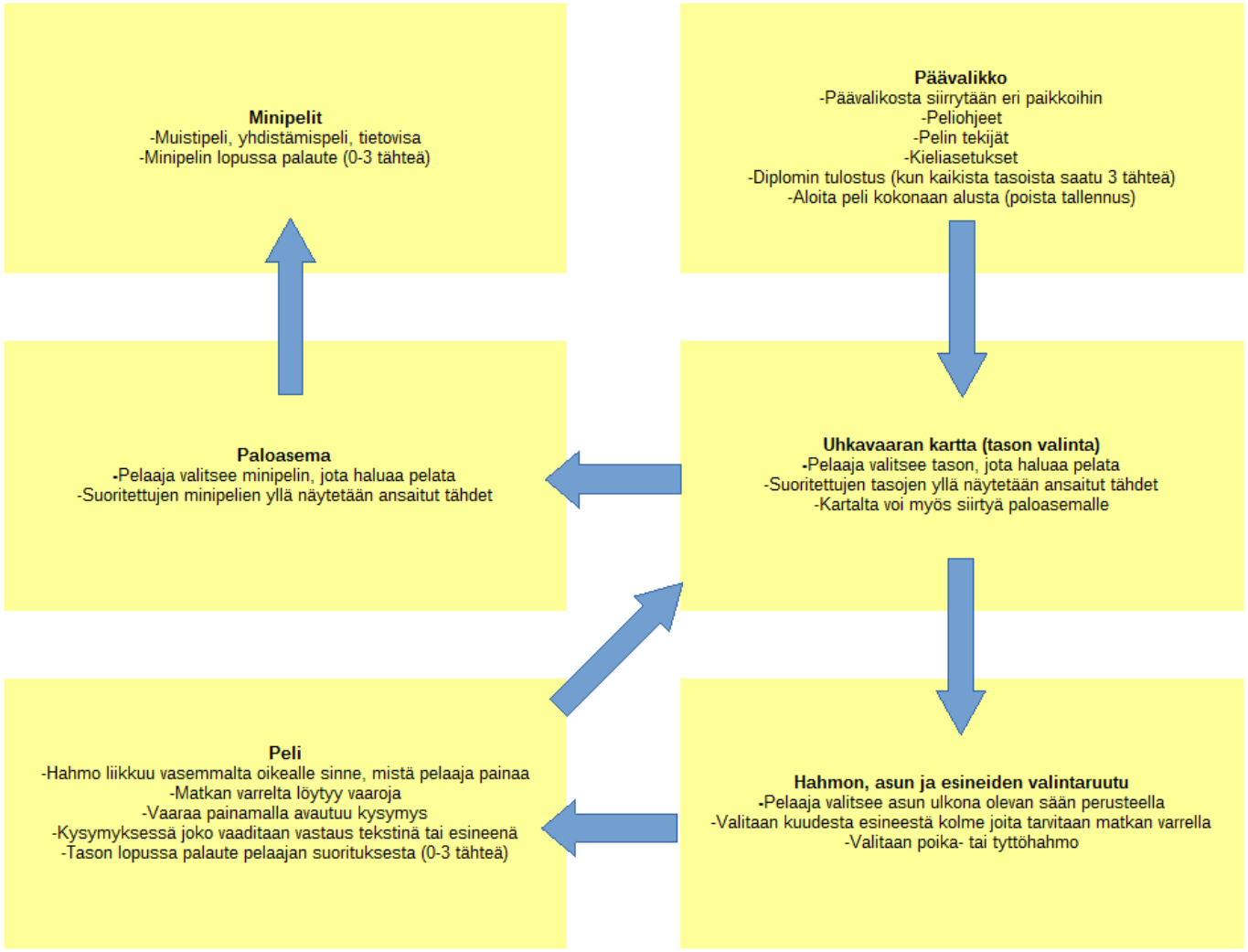
2.1 Tavoitteet

Tavoitteena oli tehdä pelistä mahdollisimman paljon asiakkaan toiveiden mukainen ja saada se AppStoreen vuoden 2015 joulukuun lopussa. Asiakas halusi lopulliseen peliin monenlaisia ominaisuuksia, kuten tulostettava diplomi, neljä eri kieliasetusta ja ääneenluku jokaiselle pelin tekstiosuudelle. Lisäksi peliin täytyi laittaa paljon tulipaloaiheisia vaaroja, sillä pelin rahoitti Palosuojelurahasto.

Pelin täytyy toimia hyvin iPadilla. Tietokoneen ruudulla pyörivä peli on hyvin erilainen iPadiin verrattuna. IPadin suoritusteho ei ole kovinkaan suuri. Pelin hyvä optimisaatio oli yksi tärkeimpiä tavoitteita.

Peli on suunnattu esikouluikäisille lapsille, joten peli ei saa olla liian vaikea. Pelissä ei ole monimutkaista ajattelua vaativia pulmia, vaan suoraan esitettyjä kysymyksiä ja yksinkertaisia minipelejä. Pelin täytyy olla tarpeeksi helppo, että jokainen saavuttaa diplomin tulostukseen vaadittavan pistemäärän.

Jokaisesta väärästä vastauksesta annetaan palaute, joka kertoo, miksi pelaaja toimi tilanteessa väärin. Myös tason lopussa annetaan palaute, jossa huomau-tetaan ohitetuista vaaroista ja annetaan pelaajalle tähtiä ansaitun pistemäärän mukaan.



Kaavio 1: Pelin eri scenet ja niiden toiminnallisuus

Yllä oleva kaavio esittää pelin eri scenejä ja niiden yhteyksiä. Laatikoissa selitetään scenen tehtävää ja nuolet näyttävät, mihin jokaisesta näytöstä pääsee.

2.2 Haasteiden tunnistaminen

Lähtokohtana on saada peli lokalisoitua usealle eri kielelle. Täytyy keksiä keino tallentaa se muotoon, josta pystyy hakemaan tarvittavan tekstikappaleen valitulla kielellä. Lisäksi jokainen teksti pelissä luetaan ääneen, joten sama on tehtävä myös äänitiedostoille. Kysymysten vastausvaihtoehdoille on myös omat kuvat, joten nekin täytyy ladata esitettävän kysymyksen mukaan.

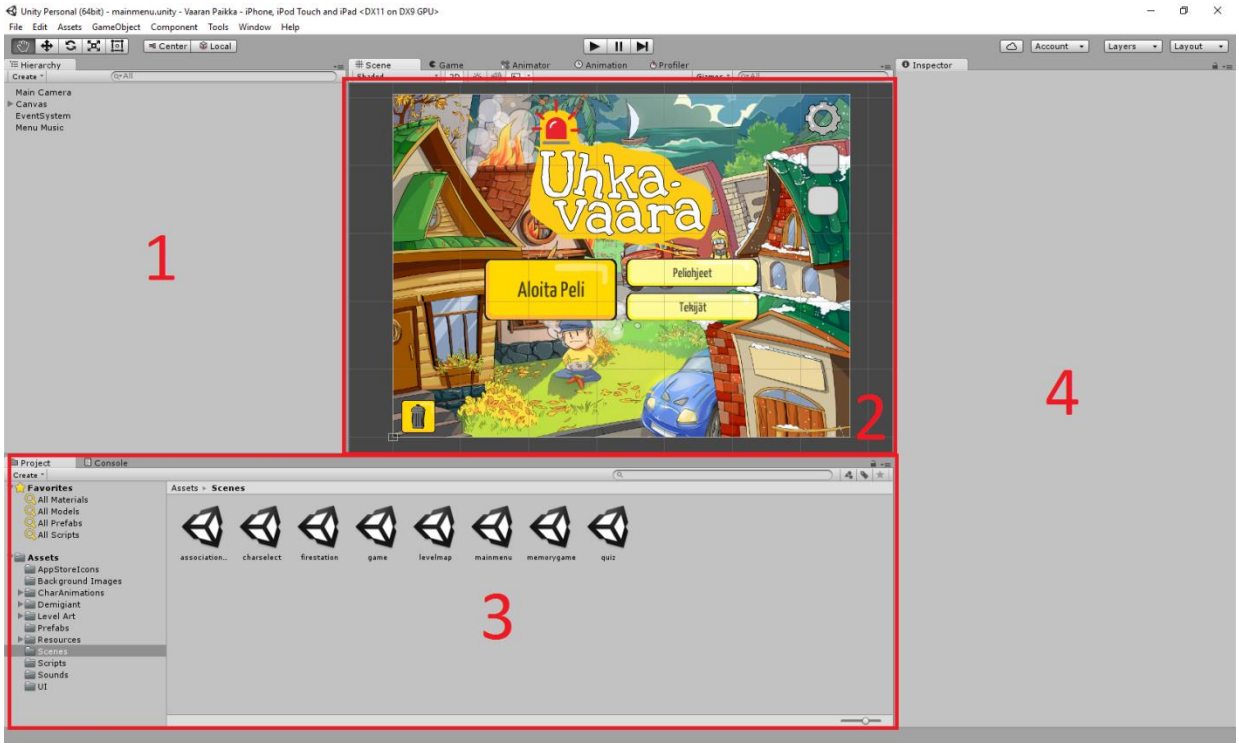
IPadista tulostaminen tuottaa todennäköisesti ongelmia. Unityssä ei ole mitään valmista ominaisuutta tulostukseen. Windowsilla tulostaminen on onnistunut avaamalla command linen kautta Paint ja tulostamaan sitä kautta, mutta iPadilla täytyy keksiä jokin toinen keino.

Muistinkäytön hallitseminen on tärkeää varsinkin iPadilla. Muistia ei ole paljon, joten peli saattaa hajota, jos tapahtuu muistivuotoja. Tekstuurit ja äänitiedostot täytyy kompressoida tehokkaasti ja tekstuurien laadun on oltava juuri tarvittavan korkea.

3 UNITY

Unity on joustava ja tehokas kehitysympäristö sekä 2D- että 3D-peleille. Unityllä luodun pelin pystyy helposti siirtämään monelle eri alustalle. (2.)

3.1 Käyttöliittymä



Kuva 1: Unityn käyttöliittymä. Avattu Scene on pelin päävalikko.

Unityn käyttöliittymä on melko yksinkertainen, ja sitä pystyy muokkaamaan haluamakseen helposti. Yllä olevassa kuvassa on eritelty Unityn käyttöliittymän eri osat.

Alue 1 on Unityn hierarchy-lista. Tässä listassa näkyvät kaikki scenen sisältämät peliobjektit. Ruudun keskellä numerolla 2 merkitty alue on ”scene”, eli alue, missä kaikki pelin elementit ovat. Aluksi sieltä löytyy vain kamera, jonka kautta pelaaja katsoo peliä, mutta projektin edetessä sinne lisätään kaikenlaisia peliobjekteja. Nämä näkyvät sitten hierarchy-listassa.

Numero 3 on projektin Assets-kansio. Tähän kansioon tai sinne tehtyihin alikansioihin laitetaan kaikki pelissä käytettävät tiedostot, kuten tekstuurit, äänet ja skriptit. Niitä voi sitten myöhemmin antaa peliobjekteille vetämällä ne Assets-kansiosta peliobjektin komponentille, jolle kyseinen tiedosto sopii.

Oikeassa reunassa numerolla 4 esitetty alue on Inspector. Tähän ikkunaan ilmestyy lisätietoja valitusta tiedostosta tai peliobjektista. Yleensä se sisältää asetuksia, kuten peliobjektin koon ja tekstuurin kompressointitason. Inspector-ikkuna näyttää kaikki peliobjektin komponentit. Jos halutaan lisätä esimerkiksi kuva pelihahmolle, se valitaan hierarchy-listasta tai scenestä ja kuva siirretään Inspectorissa näkyvälle SpriteRenderer-komponentille.

Yläreunassa ovat tärkeät napit, joilla peli käynnistetään, pysäytetään sekä siirretään peliä yksi frame eteenpäin. Niiden alla on välilehtiä, joista pystyy muuttamaan ruudun keskellä olevaa sisältöä. Game-välilehdeltä pystyy katsomaan, miltä peli näyttää scenessä olevan kameran kautta. Välilehtiä pystyy itse lisäämään. Esimerkiksi animaatiotyökalun, Asset Storen tai pelin resurssienkäyttömonitorin pystyy lisäämään.

3.2 Käyttäminen pelin rakentamiseen

Unityssä on monenlaisia ominaisuuksia. Hyvän pelin luominen vaatii eri ominaisuuksien hallintaa.

3.2.1 Scenet

Scenet sisältävät kaikki pelin objektit. Niitä voi käyttää esimerkiksi päävalikon tai tasojen luomiseen. (3.) Scenestä toiseen siirtyminen aiheuttaa pienen latauskatkon. Niitä käytetään pelin eri osien erottamiseen. Uhkavaarassa on seitsemän erilaista sceneä: Päävalikko, Tasokartta, Esineiden valinta, Peli sekä omat scenet eri minipeleille.

Scenen yläpuolella on pieni valikko, josta pystyy muuttamaan pelin mittasuhteita. Koska peli on tarkoitettu iPadille, heti projektin alussa asetettiin mittasuhteet iPadia vastaaviksi. Tämä oli erittäin tärkeää projektin kannalta, että pystyttiin asettamaan UI-elementit tarkasti suunnitelmien mukaan.

Unity on komponenttipohjainen pelimoottori, eli kaikki peliobjektit koostuvat erilaisista komponenteista. Pelin hahmo sisältää esimerkiksi SpriteRenderer-komponentin, joka piirtää hahmon grafiikan sekä Animator-komponentin, joka vaihtaa SpriteRendererin kuvaa siten, että se näyttää animaatiota. Koko peli rakennetaan peliobjekteista ja komponenteista.

3.2.2 Resurssien ja peliobjektien käsittely

Prefabit ovat peliobjektien malleja, joita voi luoda sceneen monta kappaletta. Erona peliobjektiin, joka on kopioitu sceneen monta kertaa, on se, että prefabeja pystyy muokkaamaan kaikkia kerralla. Prefabin pystyy luomaan vetämällä hierarchy-listassa olevan peliobjektin projektin Assets-kansioon.

Resurssit ovat assetteja, joita pystyy lataamaan koodissa annetulla komennolla. Resurssien käyttäminen on Unityssä erikoistapaus. Yleensä tekstuurit ja muut komponentit lisätään peliobjekteille scenessä, mutta peleissä, kuten Uhkavaarassa, joissa on niin paljon tekstiä ja ääniä, on helpompaa asettaa nämä automaattisesti koodissa. Tämän vuoksi laitettiin kaikki tekstit Resources-kansioon, josta ladataan sopiva teksti tarvittaessa. Unity ei automaattisesti tee Resources-kansiota, vaan se pitää itse luoda, mikäli sitä tarvitsee.

3.2.3 Coroutinet

Coroutinet ovat funktioita, joiden suorittamisen pystyy keskeyttämään. Niiden suorittamista pystyy sitten jatkamaan seuraavilla frameilla. (4.) Tavallinen funktio täytyy suorittaa yhden framen aikana.

Coroutinet ovat äärimmäisen tärkeitä. Esimerkkinä tietovisan palauteikkunan ääneenluku, jossa luetaan kaikkien kysymysten vastausten palautteet peräjälkeen. Tehtiin coroutine, joka lukee ensimmäisen palautteen, odottaa audiopätkän pituuden verran ja lukee seuraavan palautteen. Toistetaan, kunnes kaikki viisi on luettu. Ilman coroutineja olisi jouduttu tekemään tämä jonkin peliobjektin Update-loopissa, mikä olisi vaatinut suuren määrän if-lausekkeita, timereitä ja tarkastuksia.

3.2.4 Tweening

Tweening ei ole pelattavuuden kannalta tärkeää, mutta parantaa pelin visuaalista ilmettä huomattavasti. Tweeningillä tarkoitetaan tietyn ominaisuuden muuttamista arvosta toiseen sulavasti tietyssä ajassa. Esimerkiksi tason palauteikkunassa pelaajalle annettavat tähdet käyttävät tweeningiä. Ne luodaan niin pieninä, ettei niitä nähdä, mutta ne kasvavat yhden sekunnin aikana suuriksi ja tekevät 360-asteen käännöksen samaan aikaan. Ilman tweeningiä saman tuloksen saamiseksi olisi vaadittu yhden sekunnin pituinen ajastin, jota pienennetään jokaisella framella ja tähden Update-looppiin tarkastuksia ja tähden transform-komponentille tehtäviä toimenpiteitä. Vaihtoehtoisesti tähdet olisi voinut suoraan luoda suuriksi ilman luomisanimaatiota, mutta tämä on epämiellyttävän näköistä.

Unityssä ei ole oletuksena tweening-ominaisuutta, joten Unityn Asset Storesta ladattiin DOTween-niminen lisäosa. Se on ilmainen ja sitä saa käyttää rahallisissakin sovelluksissa luvallisesti. Lisäosa on erittäin helppokäyttöinen.

DOTween sisältää oikopolkuja tunnetuille Unityn objekteille, kuten Transformille, Rigidbodylle ja Materialille. Tweeningin pystyy aloittamaan suoraan viittaamalla näihin objekteihin (5.). Uhkavaara hyödyntää pelkästään tätä tapaa. Lisäosa tosin tarjoaa toisenkin tavan tweenata arvoja, joihin ei ole oikopolkua.

```
transform.DOScale(new Vector3(1f, 1f, 1f), 2.5f);
```

Ohjelmalistaus 1: Esimerkki tweeningin käytöstä

Yllä olevassa esimerkissä käytetään Transform-komponentilla olevaa oikopolkua aloittamaan tween, joka muuttaa sen kokoa 1 x 1 x 1-kokoiseksi 2,5 sekunnin ajan kuluessa.

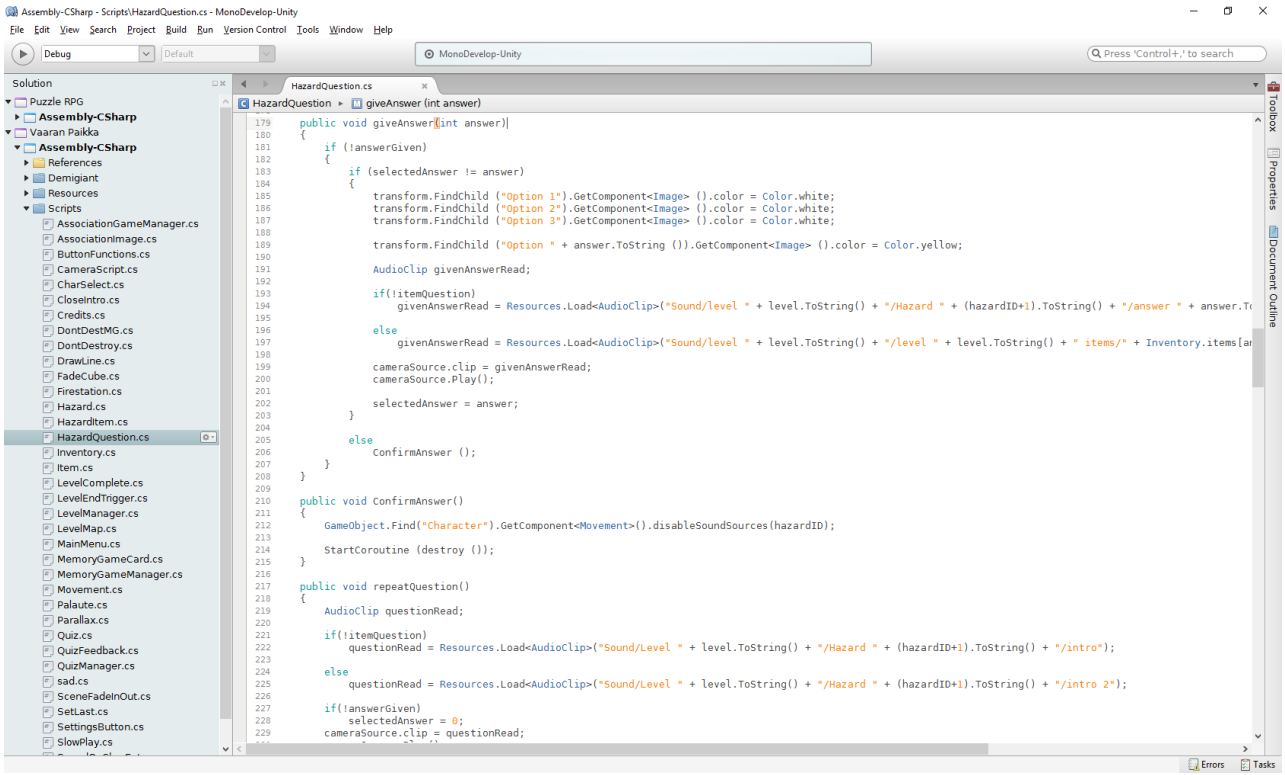
Tweeningiä käytetään pelissä monessa paikassa: vaarojen kysymysikkunoiden ilmestymisessä tyhjästä, muistipelin korttien kääntymisessä, intro-ikkunoiden sulkeutumisessa, asetusnappuloiden ilmestymisessä, hahmon jäänylitysanimaatiossa sekä monessa muussa asiassa.

3.3 MonoDevelop ja skriptit

Peliobjektit saa tekemään asioita antamalla niille skriptikomponentin. Skriptit ovat ohjelmakoodin sisältäviä tiedostoja. Unity tukee kolmea eri ohjelmointikieltä: JavaScriptiä, C#:a sekä Boota. Tämä projekti tehtiin käyttämällä C#:ia. Skripti luodaan samalla tavalla kuin mikä tahansa muukin komponentti: Assets-kansiossa hiiren oikealla painikkeella tulevasta valikosta, tai valitsemalla ”Add Component” peliobjektin inspector-valikosta.

MonoDevelop on Unityn mukana tuleva ohjelmointiympäristö. (6.) Unityssä luodut skriptit avautuvat oletuksena MonoDevelopissa, jossa niitä voi muokata. Skripteihin luodaan automaattisesti Start() ja Update()-funktiot. Startia kutsutaan silloin, kun peliobjekti luodaan. Updatea kutsutaan jokaisella framella.

Valmis skripti tallennetaan MonoDevelopin tallenna-painikkeella. Tämä päivittää skriptin myös Unityssä kaikkiin peliobjekteihin, joihin se on laitettu. Skripteihin voi laittaa julkisia muuttujia, joita pystyy muokkaamaan peliobjektin inspector-ruudulta.



Kuva 2: MonoDevelopin käyttöliittymä. Kuvassa on skripti, joka on vastuussa kysymyspeliobjektien käyttäytymisestä.

3.4 Sprite Editor ja animaatiot

2d-pelin grafiikat luodaan käyttämällä spritejä ja spritesheetejä. Kun Unityyn tuodaan jpeg- tai png-kuva, Unity tekee siitä automaattisesti sprite-tyyppisen. Spriten voi antaa peliobjektille, jolla on Sprite Renderer-komponentti. Tämä piirtää spriten ruudulle.

Kun spriteä tarkastellaan sen asetuksia pystyy muuttamaan Inspector-ruudulta. Tärkeimmät asetukset ovat Pixels Per Unit, Generate Mip Maps sekä Sprite Mode. Pixels Per Unit tarkoittaa sitä, kuinka monta pikseliä muutetaan yhdeksi Unityn mittayksiköksi. Melkein kaikki iPad-näytöt käyttävät resoluutiota 2048*1536 pikseliä. Pelini spriteihin valittiin Pixels Per Unit-arvo 102.4. Tämän vuoksi näytölle mahtuu tasan 20 pituusyksikköä leveyssuunnassa ja 15 pystysuunnassa.

Generate Mip Maps -asetuksen ollessa päällä peli näyttää heikompilaatuisia spritejä säilyttääkseen suorituskykyä muistin hinnalla. Tämä asetus kannattaa laittaa pois päältä. Se kuluttaa niin paljon muistia, että iPad lopettaa pelin suorittamisen automaattisesti, jos sitä käytetään liikaa. Asetuksen poistaminen teki pelin spriteistä myös paljon terävämpiä.

Sprite mode tarkoittaa sitä, että kuvassa on yksi vai useampia spritejä. Valitsemalla ”multiple”-sprite moden saa kuvan avattua Unityn sprite-editorissa.

Joskus on kätevää laittaa yhteen kuvaan monta eri grafiikkaelementtiä yhteen kuvaan. Unityn sprite editor -työkalun avulla pystyy helposti noutamaan kuvasta kaikki grafiikkaelementit (7.) Sprite editor on työkalu, jolla kuvan saa jaettua moniin eri spriteihin. Sprite-editorin yläkulmassa on valikko, josta pystyy jakamaan kuvan automaattisesti, pikseleiden tai haluttujen kuvien määrän mukaan. Sprite editorilla jaettuja kuvia pystyy käyttämään animaatioiden tekemiseen.

Animaation pystyy luomaan valitsemalla animaatiossa käytettävät kuvat, jotka on tehty sprite-editorilla. Vedetyt kuvat siirretään sceneen, jolloin Unity luo automaattisesti Animation Controllerin sekä kuvista tehdyn animaation. Animation Controller on komponentti, jonka avulla peliobjektille määritellään kaikki sen käyttämät animaatiot ja transitiot animaatioiden välillä.

Animation Controllerin pystyy avaamaan uuteen ikkunaan. Tämä tuo esiin tilakoneen, jossa ovat kaikki peliobjektin eri tilat eli erilaisia animaatioita vaativat tilanteet. Aluksi tilakoneessa on vain yksi tila, mutta niitä voi tarvittaessa lisätä. Pelissä on monia yhden animaation objekteja, kuten palloa pompotteleva lapsi tai palavan talon ikkunasta tuleva savu.

Tilakoneeseen luodaan uusi tila valitsemalla ”New State” hiiren oikealla painikkeella avautuvasta valikosta. Uusi tila tulee yhdistää yhteen tai useampaan vanhaan tilaan, jotta pelihahmon voi vaihtaa siihen tilaan. Tämä onnistuu tekemällä transitio. Transitiolle täytyy antaa ehto, jonka toteutuessa tila vaihdetaan. Esimerkiksi pelihahmolla se on bool-arvo, jota vaihdetaan sen mukaan, liikkuuko hahmo. Tämä ehto voi esimerkiksi olla vaatimus, että integer-arvo on tietyn suuruinen tai pelkästään se, että edellinen animaatio on suoritettu kerran. Tilakoneessa luotuja arvoja pystyy muokkaamaan koodissa ottamalla Animator-komponentti GetComponent-funktiolla.

Pelihahmolla on kaksi animaatiota: Käveleminen ja paikallaanoloanimaatio, mutta näistä on lukuisia eri versioita: jokaiselle eri asulle omat animaatiot sekä pyöräilytasossa pyöräily- ja paikallaanoloanimaatio jokaiselle asulle ja kypärän kanssa ja ilman. Lisäksi pelissä on sekä tyttö- että poikahahmo, joten määrä kerrotaan vielä kahdella. Pelihahmolla on 36 erilaista animaatiota.

3.5 Valmiin pelin viimeistely ja rakentaminen

Ennen pelin rakentamista eli projektin muuttamista päätelaitteelle ajettavaksi tiedostoiksi kannattaa käydä muuttamassa projektin asetuksia ja laittaa turhat asetukset pois päältä suorituskyvyn säästämiseksi. Varjot ja valot laitettiin kokonaan pois päältä, sillä 2d-pelissä niitä ei tarvitse.

Projektiasetusten ”Player”-alivalikosta pystyy säätämään asetuksia, jotka vaikuttavat nimenomaan pelin suorittamiseen iPadilla. Sieltä pakotetaan peli pyörimään iPadissa vaakasuorassa tasossa. myös pelin kuvake sekä splash screen -kuva asetettiin sieltä. Splash screen on kuva, joka näytetään pelin käynnistyksen yhteydessä. Myös pelin bundle identifier asetetaan täältä. Tätä tarvitaan myöhemmin Xcoden kanssa työskennellessä.

Kun peli halutaan rakentaa, valitaan File-valikosta ”Build Settings”. Avautuu ikkuna, jonne täytyy asettaa kaikki pelissä esiintyvät scenet. Tämän jälkeen valitaan käyttöjärjestelmä, jolle peli rakennetaan ja painetaan ”Build”. Tämä luo pelin build-tiedostot annettuun kansioon.

4 TOTEUTUS

Peli koostuu monesta eri scenestä. Peliin tehtiin monia valikoita, tasonvalintakartta, minipelejä sekä kenttiä.

4.1 Alkuasetukset

Uuden projektin luomisen yhteydessä käyttäjä pystyy valitsemaan haluaako käyttää 2D- vai 3D-asetuksia. Asetukset muuttavat esimerkiksi kameraa ortografisen ja perspektiivin välillä, valaistusta sekä kuvien käsittelemistä spriteinä tai tekstuureina. (8.)

Peli on kaksiulotteinen, joten peliin valittiin 2D-asetukset. Heti aluksi tehdään peliin Canvas-peliobjekti, jonka ulottuvuudet muutetaan iPadia vastaaviksi. Fysiikka-asetuksista muutetaan painovoima nollaksi, etteivät peliobjektit liiku alaspäin automaattisesti. Nyt on olemassa hyvä pohja, jonka päälle peliä voi alkaa rakentaa.

4.2 Valikot

Pelivalikot toteutettiin Unity 4.6:den mukana tulleen uudistetun UI-systeemin avulla. Sceneen luodaan Canvas-peliobjekti, johon vedetään halutut UI-elementit. Koodissa UI elementit saa tekemään haluttuja asioita, kuten lataamaan uuden scenen nappulaa painamalla tai muuttamaan peliohjeissa olevaa havainnollistavaa kuvaa sivua vaihtaessa.

UI-elementtia pystyy muuttamaan kuten mitä tahansa muutakin komponenttia. Ensin etsitään peliobjekti, jossa komponentti on käyttämällä esimerkiksi GameObject.Find()-funktiota, ja ottamalla löydetystä peliobjektista muutettava UI-elementti GetComponent()-funktiolla.

Esimerkkinä tästä koodinpätkä, joka asettaa peliohjepaneeliin tekstin riippuen siitä, mikä ohjekirjan sivuista on valittuna.

```
GameObject.Find("Instruction Text").GetComponent<Text>().text =
instructionsTexts[instructionsPage];
```

Ohjelmalistaus 2: Peliohjepaneelin tekstin muutos

Yllä olevassa esimerkissä instuctionsTexts on taulukko, joka säilöö kaikki tekstit, mitä ohjevalikossa tarvitaan, ja instuctionsPage on integer-arvo, joka vastaa valittua sivua.

Kun pelaaja siirtyy ensimmäistä kertaa kaupungin kartalle eli tasonvalintaan, hänelle näytetään pieni tarinaa alustava näyttö. Myös pelaajan suoritettua kaikki tasot hyvällä pistemäärällä näytetään tarinan päättävä ikkuna. Peli lataa pelaajan ansaitsemat tähdet eli tasojen pistemäärät aina, kun pelaaja siirtyy kartalle. Mikäli tähtiä on nolla, näytetään alkuruutu. Jos tähtiä on 15 näytetään loppuruutu. Tällöin avataan myös diplomin tulostus -vaihtoehto päävalikossa.

4.3 Peli

Pelin pääosuus eli Uhkavaaran tutkiminen oli suurin urakka. Siihen sisältyi esimerkiksi hahmon liikkuminen, vaarojen aktivointi ja kaikkien vaarojen ääni- ja tekstitiedostojen säilöminen.

4.3.1 Liikkuminen

Pelihahmo liikkuu vaakasuorasti eteenpäin osoitettuun paikkaan. Kun näyttöä painetaan, otan hiiren (tai sormen) sijainnin näytöllä Input.GetMousePosition()-funktiolla. Tämä palauttaa Vector2:n, eli luokan, joka säilöö kaksi lukua: sijainnin koordinaatiston x- ja y-akseleilla. Tätä arvoa ei voi suoraan käyttää hahmon liikuttamiseen, sillä luvut on esitetty pikseleinä näytön reunasta laskien, eikä pelimaailman koordinaatteina, joten ne on muutettava: Tähän sopii Camera-luokan funktio ScreenToWorldPoint().

Pelihahmo liikkuu eteenpäin, kunnes sen x-koordinaatti on tarpeeksi lähellä määränpäävektorin x-koordinaattia. Sijainteja verrataan Vector2.Distance()-funktiolla, joka antaa float-arvon, joka vastaa kahden pisteen etäisyyttä. Kun arvo on tarpeeksi pieni, pelihahmo pysäytetään, eli sen nopeus muutetaan nollaksi ja animaattorille annetaan käsky esittää paikallaanoloanimaatiota.

4.3.2 Pelikenttä ja vaarat

Pelikentät tehtiin luomalla aluksi tyhjän peliobjektin ja rakentamalla kentän sen päälle. Pelisuunnitteluopiskelijoiden tekemät pohjapiirrokset kentistä ladattiin kuvatiedostoina ja annettiin peliobjektin SpriteRenderer-komponentille. Tällä tavoin sain sceneen pohjan, jonka päälle pystyin laittamaan yksittäiset esineet.

Pelikentälle tehtiin myös lapsiobjekti, joka on vastuussa pelin parallax scrolling -efektistä. Tämän avulla kentän taustalla näkyvät rakennukset siirtyvät hitaammin kuin edessä olevat objektit. Tälle peliobjektille tehty skripti toimii siten, että se ottaa kameran tämänhetkisen sijainnin ja liikkuu sen x-koordinaattiin jaettuna kolmella. Parallax scrolling täytyi itse tehdä, koska pelin kamera on ortografinen eli perspektiiviä ei ole.

Vaarat ovat kuvia kuten muutkin kentässä olevat objektit. Erona on se, että painamalla vaaraa avautuu kysymysikkuna. Niille on myös annettu BoxCollider2D-komponentti. Yleensä tätä komponenttia käytetään törmäyksen tunnistamiseen, mutta sen avulla pystyy helposti tarkastamaan, onko kursori sen päällä. Kun hiiren painiketta painetaan, vaaraobjekti avaa ikkunan, jossa on kysymys. Vaara antaa sille määrätyn tunnisteen kysymysskriptille, joka tekee sen perusteella kysymyspeliobjektille oikeat tekstit, kuvat ja äänet.

Kuva 3: Pelin vaarat esitetään kysymyksinä. Joissakin vaaroissa paras vaihtoehto on tietyn esineen käyttäminen. Jos tarvittavaa esinettä ei ole, näytetään tekstivaihtoehdot.

4.3.3 Tekstin ja audion säilöminen

Pelissä on runsaasti tekstiä ja äänitiedostoja, joten täytyi keksiä keino säilöä ne siten, että pystyy hakemaan halutun tiedoston helposti. Teksti kopiotiin tekstitiedostoon, ja muokattiin sitä erottamalla eri vaarojen tekstikappaleet tietyllä merkillä. Kysymykset ja vastaukset erotettiin toisenlaisella merkillä. Projektin Assets-kansioon luotiin uusi kansio, joka nimettiin ”Resources”. Unity pystyy hakemaan tämännimisestä kansiosta tiedostoja Resources.Load()-funktiolla.

Koodissa noudetaan tekstitiedosto, kun pelaaja avaa kysymyksen. Tekstitiedosto jaetaan viiteen eri kappaleeseen käyttämällä Split()-funktiota. Tämän avulla pystytään erottamaan tekstikappaleen tiedostoon ujutettujen merkkien avulla. Haluttu kappale jaetaan edelleen kysymykseen ja kolmeen vastausvaihtoehtoon samalla tavalla. Viimeiseksi lisätään tekstin kysymyspeliobjekti tekstikomponentteihin.

Audiotiedosto laitetaan samalla tavalla Resources-kansioon, josta ne ladataan ja soitetaan kysymyksen ilmestyessä ruudulle.

4.3.4 Kentän läpäiseminen

Kentän loppupäässä on triggeri, joka avaa tason palauteruudun, kun pelaaja kävelee sen yli. Palauteruutu arvioi pelaajan suorituksen antamalla hänelle nollasta kolmeen tähteä ansaitun pistemäärän mukaan. Pisteitä kerätään vastaamalla oikein vaarakysymyksiin ja valitsemalla oikeat esineet ja tason vaatima asu.

Kuva 4: Palauteikkuna

Palauteruudussa on jokaista vaaraa vastaava kuva, jonka päällä on hyväksytty- tai hylätty-merkki riippuen siitä kuinka pelaaja on kyseiseen vaaraan vastannut. Vaaran läpäiseminen tallettaa kyseisen vaaran suorituksen boolean-taulukkoon, josta kentän lopussa katsotaan kumpi merkki suorituslaatikkoon ladataan.

Palauteteksti ladataan sen mukaan, kuinka monta pistettä pelaaja on saanut. Tekstit ladataan kuten kysymyksetkin: Resources-kansiosta. Pelaajan saavuttaman pistemäärän perusteella pelaajalle annetaan myös tähtiä. Tähdet ilmestyvät palauteruudulle yksi toisensa jälkeen tweening-efektiä käyttäen. Tämä parantaa palauteikkunan visuaalista ilmettä huomattavasti.

4.4 Minipelit

Paloasema sisältää kolme erilaista minipeliä: Tietovisan, muistipelin sekä yhdistämispelin. Nämä minipelit tuovat peliin vaihtelua ja erilaista sisältöä kaupungin tutkimisen lisäksi.

4.4.1 Tietovisa

Tietovisassa esitetään pelaajalle kysymyksiä, kuten pelin vaaroissakin. Tämän vuoksi käytin pohjana vaarojen yhteydessä esitettävän kysymyksen prefabia. Kysymyksiä on yhteensä 10, joista esitetään 5 satunnaisesti. Toteutin tämän tallettamalla taulukkoon luvut yhdestä kymmeneen, sekoittamalla lukujen järjestyksen, ja valitsemalla sekoitetusta taulukosta viisi ensimmäistä lukua. Näiden lukujen mukaan valitaan kysymys.

Kysymykset tallennettiin tekstitiedostona Resources-kansioon. Ne noudetaan koodissa käyttämällä Resources.Load()-funktiota, jonka jälkeen jaetaan noudettu tekstipätkä kymmeneen kysymykseen erottamalla ne tekstitiedostoon laitetuilla merkillä.

Esimerkki toteutuksesta:

```
string teksti = ”kissa*koira”;
string[] tekstitaulukko = teksti.Split('*');
```

Tekstitaulukkoon saadaan täten kaksi erillistä tekstipätkää: ”kissa” ja ”koira”.

Kuva 5: Tietovisa on samanlainen pelissä esiintyvien vaarojen kanssa.

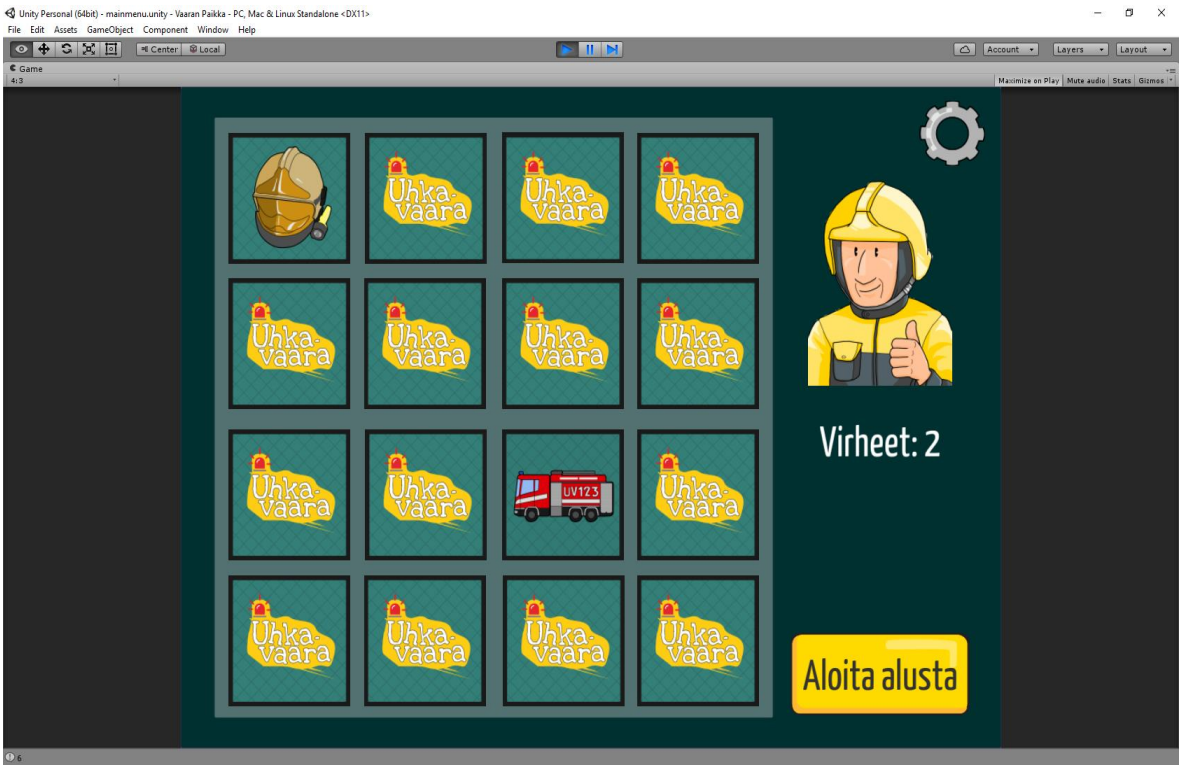
4.4.2 Muistipeli

Muistipelin tekemisessä haasteena oli luoda kortteja, joihin ladataan kuva niiden kääntyessä. Sen jälkeen pitää vielä vertailla, ovatko kuvat samanlaisia.

Tekeminen aloitettiin tallettamalla kuvat Resources-kansioon. Täältä ne voi ladata tarvittaessa. Tämän jälkeen luodaan ruudukko, jossa on 16 korttia neliömuodostelmassa.

Muistipeliin valitaan kahdeksan kuvaparia yhteensä kymmenestä. Toteutus on samanlainen tietovisan kanssa: taulukkoon laitetaan kymmenen lukua, taulukko sekoitetaan ja valitaan kahdeksan ensimmäistä lukua. Lukujen avulla ladataan kortteihin kuvat Resources.Load()-funktiolla. Muistipelissä tosin kuvat ladataan vasta kortteja kääntäessä.

Korttien kääntämisen toteutettiin lataamalla Unityyn HOTween-lisäosan. Tämä lisäosa tuo Unityyn helppokäyttöisen ”tweening”-ominaisuuden, eli peliobjektien ominaisuuksia pystyy sulavasti muuttamaan. Esimerkiksi tason suorituksen yhteydessä esitettävässä palauteruudussa olevat tähdet ilmestyvät näytölle kasvamalla pienestä suureksi. Muistipelissä tweeningin avulla pystyy helposti kääntämään kortit ympäri.



Kuva 6: Kuvaa muistipelistä

4.4.3 Yhdistämispeli

Yhdistämispeli oli minipeleistä haastavin toteuttaa. Pelissä kentälle luodaan kymmenen esinettä, jotka yhdistetään vetämällä viiva kahden sopivan esineen välille. Tämä aiheutti monenlaisia haasteita: Pitää tarkastaa kahden esineen sopivuus, soittaa esineiden ääneenluku, ja piirtää viiva esineiden välille.

Yhdistämispeliin valitaan 10 esinettä eli viisi paria. Yhteensä mahdollisia esineitä on 20. Näistä valitaan satunnaisesti peliin viisi eri paria. Kuten muistipelissäkin, tallensin esineiden kuvat Resources-kansioon. Sieltä saa helposti halutun kuvan Resources.Load()-funktiolla. Koodiin luotiin int-taulukko, joka vastaa valittuja pareja. Taulukkoon arvotaan 10 satunnaista numeroa. Sitten taulukko sekoitetaan, ja otetaan 5 ensimmäistä numeroa. Näiden lukujen mukaan ladataan vastaava kuva Resources-kansiosta ja siirretään peliobjektiin.

Viivan piirtäminen ei ollut niin helppoa kuin aluksi kuviteltiin. Unityssä valmiiksi olevalla LineRenderer-luokalla pystyy helposti piirtämään viivan mallisen peliobjektin. Tämä luokka ei kuitenkaan ollut yhteensopiva uudistetun UI-systeemin kanssa, joten täytyi keksiä jotakin muuta. Päädyttiin käyttämään UI-systeemiin tarkoitettuja kuvakomponentteja. Kuvaa sitten venytetään, käännetään ja siirretään siten, että se näyttää alku- ja nykyisen pisteen välillä olevalta viivalta.

Kuva 7: Yhdistämispeli. Pelaaja vetää viivaa palavista vaatteista.

4.5 Kieliasetukset

Lopullisessa pelissä on ainakin kaksi eri kieliasetusta: Suomi ja ruotsi. Lisäksi mahdollisesti englanti ja venäjä. Vaikka kieliominaisuutta ei vielä pelissä ole, tapa sen toteuttamiseksi on valmiiksi suunniteltu.

Jonnekin tehdään muuttuja, jonka arvon saa helposti haettua joka paikasta. Tämän vuoksi siitä tehdään public static -tyyppisen. Muuttuja voi olla mikä tahansa, kuten integer, string tai char. Tämä muuttuja vastaa valittua kieliasetusta, jota pystyy vaihtamaan päävalikosta.

Erikieliset ääni- ja tekstitiedostot laitetaan samaan paikkaan kuin suomenkielisetkin. Niiden nimi on sama, paitsi niiden perässä on kielen tunniste, jonka avulla ne pystyy lataamaan muuttujaa käyttäen.

Esimerkiksi ruotsin kieltä voi esittää string-arvo ”swe”. Hazards1 on ensimmäisen tason vaarojen tekstitiedoston nimi suomeksi. Ruotsin kielieä vastaava nimetään Hazards1swe:ksi ja koodissa ladataan käyttämällä seuraavaa metodia:

```
Resources.load<TextAsset>(”Hazards1” + language);
```

Ohjelmalistaus 3: Esimerkki resurssien lataamisesta

Yllä olevassa esimerkissä language on kieltä vastaava string-muuttuja. Suomen kieltä vastaisi tyhjä string-arvo, ettei minun tarvitsisi nimetä uudelleen suomenkielisiä tiedostoja.

4.6 Muut pelielementit

Äänet on eroteltu sekä peliääniin että ääneenlukuun. Molemmille on omat säätimet, joilla ne saa laitettua pois päältä. Pelin Audio Sourcet merkittiin käyttämällä Unityn tag-systeemiä sen perusteella, kumpaan ääniryhmään se kuuluu. Kun pelaaja painaa nappulaa, tekemäni funktio käy läpi kaikki pelin ääneksi tagatut peliobjektit ja hiljentää sopivat. Ääneenluku- ja peliääninappuloihin ladataan kuvat käyttäen Resources-kansiota sen mukaan, onko kyseinen asetus päällä vai ei. Molemmat asetukset on tallennettu public static booleaniin. Staattiset arvot pysyvät samoina, vaikka pelin sceneä vaihdetaan.

Hahmon ja esineiden valinta toteutettiin kokonaan UI:na. Taustalla on koko näytön kokoinen kuva ja edessä muut elementit. Kuvien sijaintia eteen- ja taakse-suunnassa pystyy vaihtamaan siirtämällä elementtien paikkoja Unityn hierarchy-listassa.

Hahmon kantamat esineet eli inventory on tehty tekemällä Esine-luokka, jossa on ominaisuudet kuten esineen nimi, kuvaus sekä esineen kuva. Hahmon liikkuminen- luokassa tehtiin Esine-tyyppisen taulukko, johon pelaajan esineet tallennetaan. Inventory olisi luultavasti ollut helpompi toteuttaa käyttämällä List-tyyppistä kokoelmaa. Taulukkototeutuksessa täytyi esineen poistamisen yhteydessä siirtää muita elementtejä takaisinpäin tyhjiin taulukon alkioihin. Tämä aiheuttikin hieman ongelmia kun ilmeni, että esineiden poistaminen ei siirtänytkään kunnolla taulukon viimeistä esinettä. Myöhemmin bugi korjattiin, mutta se jäi pelin demoversioon, joka oli kaksi kuukautta AppStoressa.

5 XCODE

Xcode on kokonaisuus, jonka avulla pystyy luomaan ohjelmia Applen laitteille. Se sisältää Xcode-kehitysympäristön, Objective-C-kääntäjiä ja erilaisia SDK:ita. (9.)

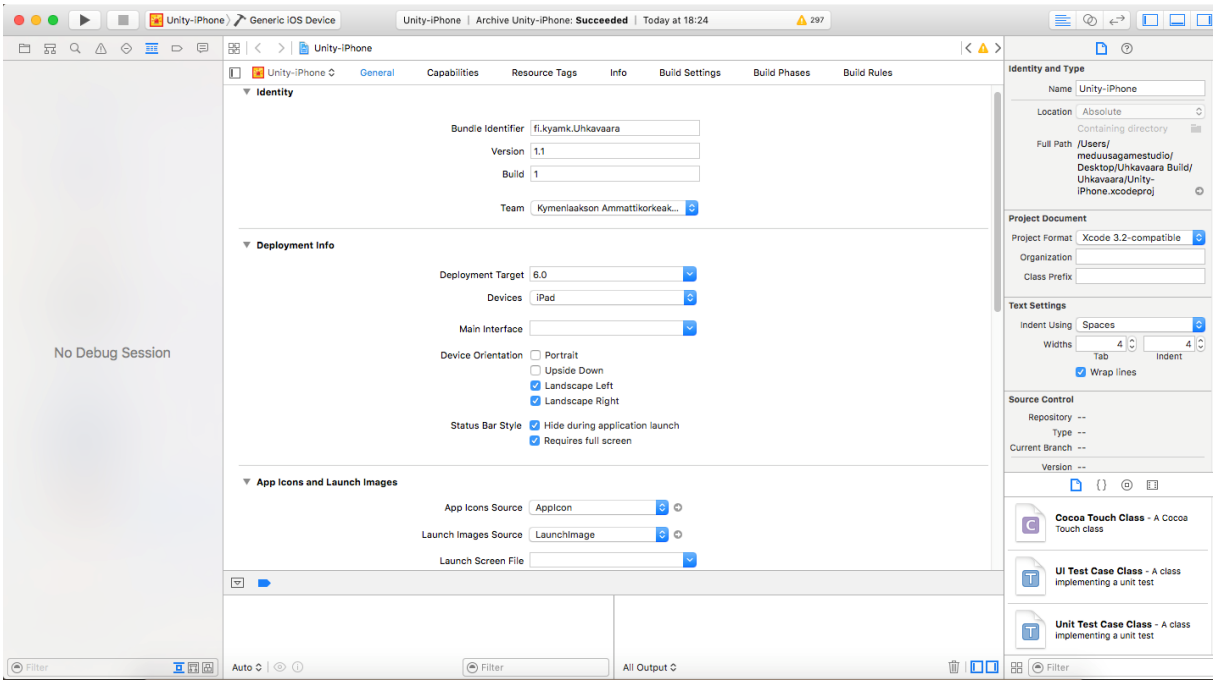
5.1 Pelin testaaminen

Jotta pelin saa siirrettyä iPadille, se täytyy tehdä Xcoden kautta. Xcode on Applen kehittämä ohjelmointiympäristö, jonka avulla voi luoda ohjelmia Applen laitteille. Unity luo pelistä automaattisesti Xcode-projektin, mutta iPadille pelin saamiseen on tästä vielä pitkä matka.

Xcoden käyttämiseen vaaditaan lisenssi, jonka saa pyydettyä vain Applen laitteella. Tämän vuoksi esimerkiksi Windows-käyttöjärjestelmällä ei pysty siirtämään peliä iPadille.

Xcode-projekti avataan Unityn luomasta projektikansiosta. Xcodessa näkyy pelin asetukset, joista suurin osa on määritelty Unityssä. Tärkeimpiä ovat bundle identifier ja team.

Bundle identifier on ohjelmalle annettava tunniste, jonka avulla eri järjestelmät tunnistavat sen. Esimerkiksi pelissä tehtävät mikromaksut käyttävät bundle identifieriä tunnistaakseen ohjelman (10.). Bundle identifier on kuin käänteinen verkko-osoite. Uhkavaaran bundle identifier on fi.kyamk.Uhkavaara.



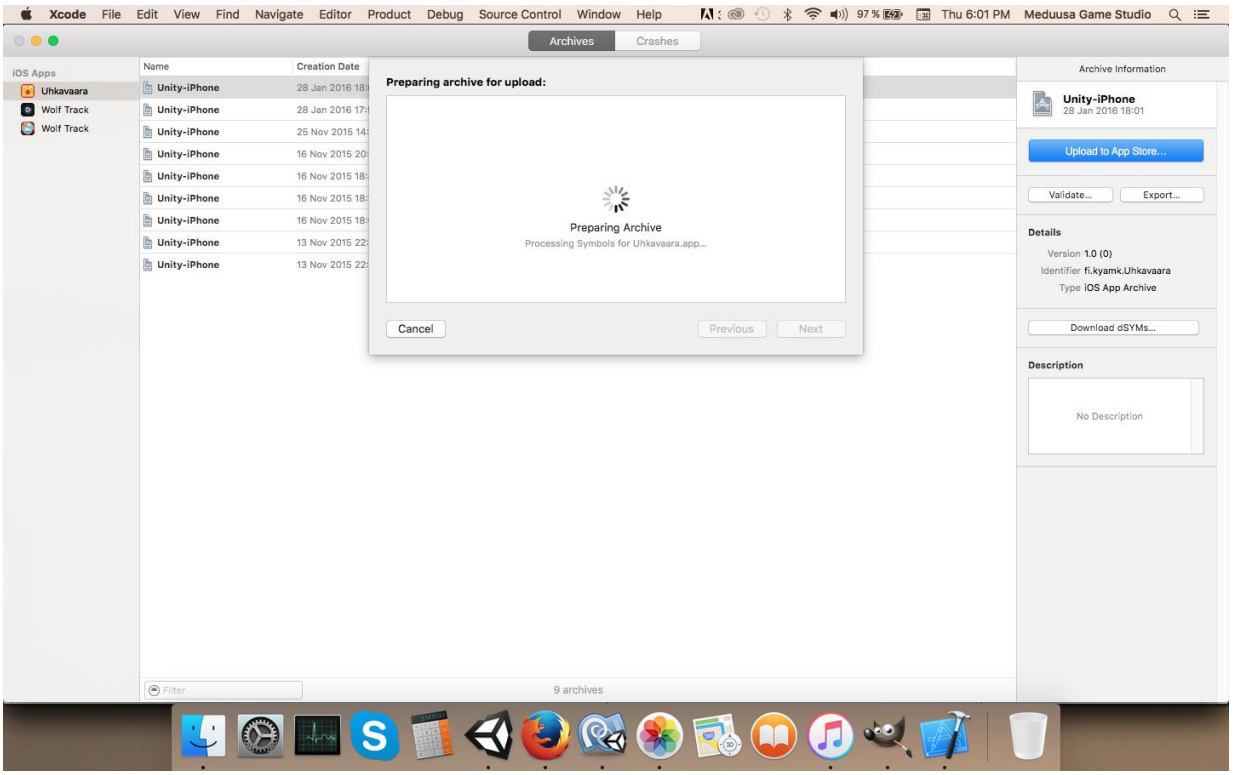
Kuva 8: Tämä näkymä avautuu, kun Unityn luoman Xcode-projektin avaa. Täältä muutetaan monenlaisia asetuksia.

Team eli tiimi on ryhmä, joka koostuu yhdestä tai useammasta henkilöstä joilla on jokin rooli pelin luomisessa. Team-tili säilöö profiileita ja sertifikaatteja, joita app vaatii. Itse en ollut tiimin luomisessa läsnä, vaan minun täytyi vain valita Xcoden listasta valmiiksi luotu tiimi.

5.2 Pelin lataaminen AppStoreen

Kun peli halutaan laittaa AppStoreen, siitä täytyy ensin tehdä arkisto. Arkisto tehdään joka ohjelmalle jakelutavasta huolimatta. Xcode-arkistot säilyttävät ohjelman sekä tärkeät debug-tiedot Xcoden hallitsemassa paketissa. (11.)

Tämä luodaan Xcoden kautta. Valittu arkisto voidaan sitten ladata AppStoreen painamalla ”Upload to AppStore”-nappulaa, joka näkyy valitun arkiston vieressä Xcoden Organizer-ikkunassa.



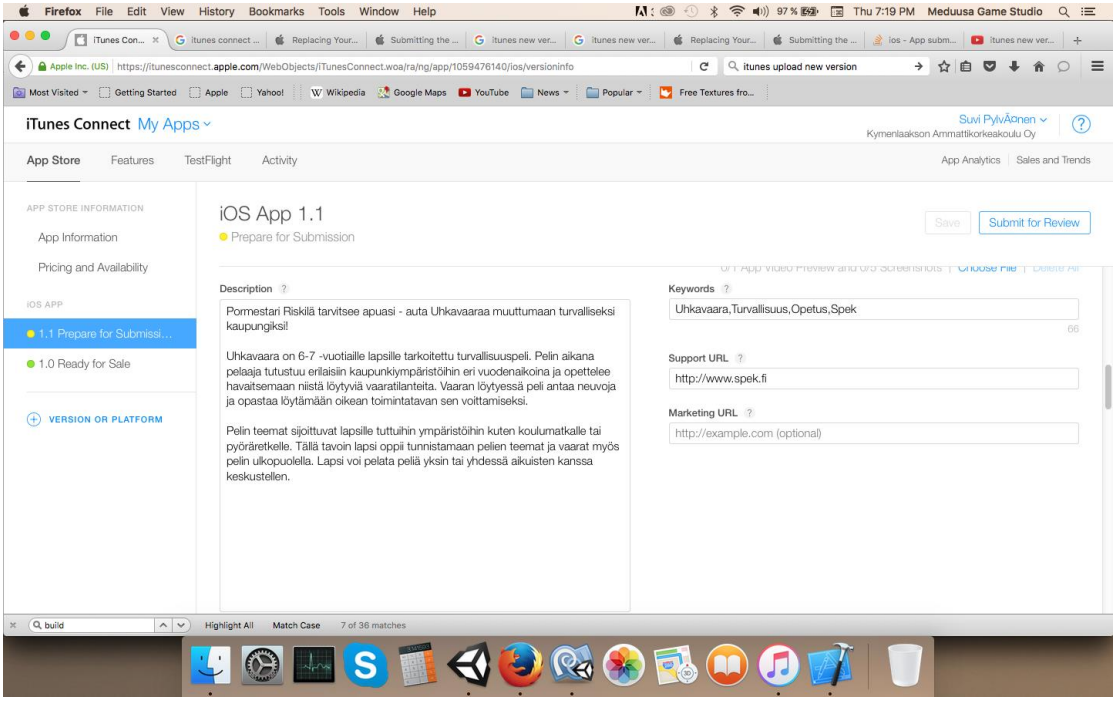
Kuva 9: Xcoden arkisto-ikkuna. Jokainen pelistä luotu arkisto näkyy listassa. Uusinta arkistoa yritetään ladata AppStoreen.

Matka AppStoreen ei tämän projektin tapauksessa kuitenkaan ollut mutkaton. Xcoden ladatessa projektia ilmaantui useita virheitä. Esimerkiksi annetuilla Apple Developer ID-tileissä ei ollut tarpeeksi oikeuksia pelin lataamiseen.

Ennen kuin pelin pystyy lataamaan AppStoreen, täytyy luoda iTunes Connectin kautta pelille oma sivu. Applen servereille ladatut peliversiot näkyvät tällä sivulla myöhemmin.

5.3 Viimeiset askeleet

Kun peli on viimein ladattu AppStoreen, täytyy vielä kirjautua iTunes Connectiin ja asettaa pelin AppStore-sivulle tarvittavat tiedot, kuten kuvaus, ikärajoitus ja kuvakaappauksia.



Kuva 10: Kuva iTunes Connectista. Täältä asetetaan tiedot, jotka näkyvät pelin AppStore-sivulla.

Xcoden kautta Applen servereille ladatut versiot pelistä näkyvät iTunes Connectissa. Sivulla pystyy valitsemaan, mikä versio pelistä AppStoreen laitetaan.

Kun nämä vaiheet on tehty, pelin voi lähettää arvioitavaksi Applelle. Tämä on viimeinen este ennen kuin pelin saa AppStoreen. Apple tarkastaa kaikki AppStoreen tulevat ohjelmat. Mikäli peli ei täytä Applen vaatimuksia se hylätään ja virheet täytyy korjata ennen uutta yritystä. Yleisimpiä hylkäyksen syitä ovat esimerkiksi bugit ja crashit, placeholder-assetit ja kehno käyttöliittymä. (12.)

6 LOPPUTULOS

Pelistä tuli loppujen lopuksi odotusten mukainen. Kaikki ominaisuudet, mitkä pelissä oli tarkoitus olla, pystyin tekemään. Matkan varrella ilmaantui useitakin ongelmia, mutta selvisin niistä etsimällä ratkaisuja internetistä tai kokeilemalla erilaisia toteutustapoja.

Projekti opetti paljon uusia asioita. Xcoden käyttämisestä ei ollut aiempaa kokemusta. Monen tunnin kamppailun jälkeen opin pelin siirtämisen iPadille ja AppStoreen hyvin. Prosessi olisi kuitenkin erittäin monimutkainen, jos lisenssit ja tilit täytyisi hankkia itse.

Kuva 11: Valmis peli AppStoressa

Parannettavaa pelissä on ainoastaan muistinkäyttö. Tekstuureja ja ääniä kompressoitiin mahdollisimman paljon, mutta niiden lataaminen kuluttaa silti runsaasti muistia. Joissakin osuuksissa ilmaantuu pientä tökkimistä, mutta muuten peli pyörii hyvin. Pelin suorituskykyä pystyisi ehkä lisäämään lataamalla assetteja aikana, jolloin pelaaja ei huomaa sitä.

LÄHTEET

1. Kaakkois-Suomen Pelastusalanliitto ry. Etusivu. http://www.kaspeli.fi/fi/etusivu [viitattu 12.2.2016]

2. Unity Technologies. Multiplatform. http://unity3d.com/unity/multiplatform [viitattu 1.2.2016]

3. Unity Technologies. Scenes http://docs.unity3d.com/Manual/CreatingScenes.html [viitattu 15.3.2016]

4. Unity Technologies. Coroutines. http://docs.unity3d.com/Manual/Coroutines.html [viitattu 7.3.2016]

5. Daniele Giardini. DOTween – Documentation http://dotween.demigiant.com/documentation.php [viitattu 21.1.2016]

6. Unity Technologies. MonoDevelop. http://docs.unity3d.com/Manual/MonoDevelop.html [viitattu 26.2.2016]

7. Unity Technologies. Sprite Editor. http://docs.unity3d.com/Manual/SpriteEditor.html [viitattu 21.1.2016]

8. Unity Technologies. 2D and 3D Mode Settings http://docs.unity3d.com/Manual/2DAnd3DModeSettings.html [viitattu 23.1.2016]

9. Apple Inc. Xcode on the Mac App Store https://itunes.apple.com/us/app/xcode/id497799835 [viitattu 7.2.2016]

10. Apple Inc. Configuring Your Xcode Project for Distribution. 2016. https://developer.apple.com/library/ios/documentation/IDEs/Conceptual/AppDistributionGuide/ConfiguringYourApp/ConfiguringYourApp.html [viitattu 18.1.2016]

11. Apple Inc. 2016. Uploading Your App to iTunes Connect https://developer.apple.com/library/ios/documentation/IDEs/Conceptual/AppDistributionGuide/UploadingYourApptoiTunesConnect/UploadingYourApptoiTunesConnect.html [viitattu 26.2.2016]

12. Apple Inc. Common App Rejections https://developer.apple.com/app-store/review/rejections/ [viitattu 1.2.2016]